Setting up models

In the folder `/Examples` we provide some showcase applications that can be used as a starting point for new modeling projects.

Section 1 describes how a model can be set up and Section 2 how experimental data can be linked to the model.

Please note that the software make heavy use of identifiers for all model and data quantities, for instance for dynamic variables, inputs, parameters, observables, ... It is very important to keep these identifiers consistent throughout one project.

Table of Contents

1. Model Definition File

A model definition file is a plain text file with the ending `.def` located in the subfolder `/Models` of the current MATLAB working directory. The illustration used in the following stem from the example applications in the `/examples` subfolder of the main code folder: `Becker_2010` and `Bachmann_2011`. The model definition files are structured in various section:

```
DESCRIPTION
```

```
...

PREDICTOR
...

COMPARTMENTS
...

STATES
...

INPUTS
...

REACTIONS (or ODES)
...

DERIVED
...

OBSERVABLES
...

ERRORS
...

CONDITIONS
...
```

Each section will be explained in the following. Lines in the model definition file can be commented out using `//`.

1.1 DESCRIPTION section

Can be used to store meta information about the model as quoted lines of text, e.g.

```
DESCRIPTION
"Model of Epo receptor internalization"
"Assembled by RG Timmer & RG Klingmüller"
"Date: 2010"
"Version: 1.0"
```

1.2 PREDICTOR section

Is used to define the independent variable / the predictor. For dynamics model, this is usually time, e.g.

```
PREDICTOR
t                T    "min"     "time"    0    100
```

- The 1st argument specifies the unique identifier (here `t`) that can be used in the mathematical expressions later.
- The 2nd-4th argument specify the unit type (here `T` = time), the unit itself

(here `"min"`), and a label used for plots (here `"time"`).
- The 5th and 6th argument specify the range of the independent variable (here 0-100 minutes).

1.3 Defining COMPARTMENTS

Can be used to define compartments in a model, for instance cytosol and nuclear compartment in the cell

```
COMPARTMENTS
cyt             V   "pl"      "vol."        0.4
nuc             V   "pl"      "vol."        0.275
...
```

The concentration dynamics of the model, in particular the transport reactions, are solved with respect to the size of the compartments.

- The 1st argument specifies the unique identifiers (here `cyt` and `nuc`) that can be used in the mathematical expressions later.
- The 2nd-4th argument specify the unit types (here `V` = volume), the units itself (here `"pl"`), and labels used for plots (here `"vol."`).
- The 5th argument specify the numerical size of the compartments

If no compartments are required, the section should be left empty.

1.4 Defining the dynamic STATES of the model

This section describes the dynamic variables / states that evolve over time as will be specified in the REACTIONS / ODES section later.

```
STATES
STAT5           C   "nM"      "conc."       cyt 1 "STAT5"                 1
pSTAT5          C   "nM"      "conc."       cyt 1 "phospho STAT5"         1
npSTAT5         C   "nM"      "conc."       nuc 1 "nuclear phospho STAT5" 1
...
```

- The 1st argument specifies the unique identifiers (here `STAT5`, `pSTAT5` and `npSTAT5`) that can be used in the mathematical expressions (reactions / ODE system) later. Initial conditions for each dynamic variable will be generated as free parameters using the prefix `init_`.
- The 2nd-4th argument specify the unit types (here `C` = concentration), the unit itself (here `"nM"`), and a plain text used for plots (here`"conc."`).
- The 5th argument specify the compartment where the state lives in (here either `cyt` or `nuc`). This can be left empty if no compartments were specified above.
- The 6th argument is a flag if the state should be showed in plots (0=no, 1=yes), default is yes.
- The 7th argument a clear text label for plotting.
- The 8th argument specifies if the state is strictly positive (0=no, 1=yes), default is yes.

1.5 Defining the INPUTS

In this section input function, usually depending on the independent variable, can be specified. The input function can be used in the mathematical expression later, such as the reaction rate equations or the observables. Please note that the input do not belong to a specific compartment. The user has to ensure plausibility of the usage of the input function himself.

```
INPUTS
Epo             C   "units/cell"   "conc."   "k1*exp(-k2*t)"
SAv             C   "units/cell"   "conc."   "k3"
...
```

- The 1st argument specifies the unique identifiers (here `Epo` and `SAv`) that can be used in the mathematical expressions later.
- The 2nd-4th argument specify the unit types (here `C` = concentration), the unit itself (here `"units/cell"`), and a plain text used for plots (here `"conc."`).
- The 5th argument specify the mathematical expression of the input function (here `"k1*exp(-k2*t)"` is an exponential decay and `"k3"` is a constant with unknown value).

Sometimes the input function should account for a change in the value of a dynamic variable, see the bolus injection example.

1.5.1 Step functions, cubic splines and user defined functions

Step functions, splines and user defined functions can be used as mathematical expression of the input function.

Step functions can be defined by:

- `"step1(t, level1, switch_time, level2)"` giving a simple step function at the value `switch_time` of the independent variable (usually time).
- `"step2(t, level1, switch_time1, level2, switch_time2, level3)"` giving a double step function at the values `switch_time1/2` of the independent variable `t`.

Cubic splines with 3, 4 or 5 knots can be defined by:

- `"spline3(t, t_knot1, p_knot1, t_knot2, p_knot2, t_knot3, p_knot3, q_initial_slope_constraint, initial_slope)"`
- `"spline4(t, t_knot1, p_knot1, t_knot2, p_knot2, t_knot3, p_knot3, t_knot4, p_knot4, q_initial_slope_constraint, initial_slope)"`
- `"spline5(t, t_knot1, p_knot1, t_knot2, p_knot2, t_knot3, p_knot3, t_knot4, p_knot4, t_knot5, p_knot5, q_initial_slope_constraint, initial_slope)"`

Here, `t` denotes the independent variable, `t_knot1-5` indicate the locations of the spline knots (these should be fixed to a numeric value),`p_knot1-5` denote the spline parameters (these can either be fixed or left as free parameters to be estimated) and`q_initial_slope_constraint` is a flag (`0` = no, `1` = yes) that indicates if the slope of the spline at the first knot is to be constained to the value given in `initial_slope` (both values should be fixed to a numeric value). The spline parameter p_knot is defined as the value u(t_knot) of the spline at t_knot. If the spline should be constrained to positive values use the functions `spline_pos3`, `spline_pos4` and `spline_pos5` the same way as described above. An example using splines is given in `Examples/Swameye_PNAS2003`.

Custom input function can be defined in the c-files `arInputFunctionsC.c` and `arInputFunctionsC.h`.

1.5.2 A general input function

Often, an input consists of transient and sustained parts. Such behaviour can be implemented by the following expression:

```
"gif_amp_trans*(1-exp(-t/gif_timescale_sust))*exp(-t/(gif_timescale_trans)) + gif_amp_
```

The function has three parameters, two amplitudes (`gif_amp_trans` and `gif_amp_sust`) and two time scales (`gif_timescale_trans` and `gif_timescale_sust`), that encode the transient and sustained parts.

1.6 Defining the dynamics

This section describes the ODE system that determines the time evolution of the states. One can either specify `REACTIONS` or `ODES` directly.

1.6.1 Using REACTIONS

The ODE system can be specified as biochemical reaction network.

```
REACTIONS
Epo + EpoR      ->  Epo_EpoR    CUSTOM "k1 * Epo * EpoR"    "label1"
STAT5           ->  pSTAT5      CUSTOM "k2 * STAT5"         "label2"
pSTAT5          ->  npSTAT5     CUSTOM "k3 * pSTAT5"        "label3"
...
```

- The 1st argument describes the reaction in a simple notation (here `Epo + EpoR -> Epo_EpoR` is a complex formation, `STAT5 -> pSTAT5` a protein modification, and `pSTAT5 -> npSTAT5` a transport reaction between compartments).
- The 2nd argument is a keyword that specifies the reaction type, `CUSTOM` or `MASSACTION`. **`CUSTOM` is the recommended usage.**
- In case of a `CUSTOM` reaction, the 3rd argument specifies the mathematical expression of the reaction rate equation (here `"k1 * Epo * EpoR"`, `"k2 * STAT5"` and `"k3 * pSTAT5"`). In case of a `MASSACTION` reaction, the mathematical expression in the 3rd argument will be just one parameter `name` that will be the rate constant corresponding to this reaction. In case of a `MASSACTION` reaction, one can use `<->` in the 1st argument to specify a reversible reaction. In this case, there will be two parameters with suffix `name_1` for the forward and `name_2` for the back reaction.
- The 4th argument (optional) specifies a label for the reaction used for plotting.

1.6.2 Using ODES

In contrast to the REACTIONS notation, the ordinary differential equations can be specified directly:

```
ODES
"mathematical expression 1"
"mathematical expression 2"
"mathematical expression 3"
...
```

where these mathematical expressions define the right-hand sides of the ODEs. Take care that there are as many ODEs as dynamic variables in the model.

1.7 Defining DERIVED variables

In this section variable that are derived from the dynamic and input variables can be defined.

```
DERIVED
Epo_ext          C   "pM"   "conc."   "Epo + dEpo_e"
Epo_int          C   "pM"   "conc."   "Epo_EpoR_i + dEpo_i"
...
```

- The 1st argument specifies the unique identifiers (here `Epo_ext` and `Epo_int`) that can be used in the mathematical expressions later.
- The 2nd-4th argument specify the unit types (here `C` = concentration), the unit itself (here `"pM"`), and a plain text used for plots (here `"conc."`).
- The 5th argument specify the mathematical expression of the input function (here `"Epo + dEpo_e"` and `"Epo_EpoR_i + dEpo_i"`).

1.8 Defining the OBSERVABLES

At this point of the model definition files one can specify the default model observables and their corresponding error model (Section 1.9). While the section in the model definition file are optional the corresponding sections in the data definition file (see Section 2.3) are mandatory and override the default specifications given here.

```
OBSERVABLES
tSTAT5_au          C   "au"  "conc."   1   1   "scale_tSTAT5 * (STAT5 + pSTAT5)"
pSTAT5_au          C   "au"  "conc."   1   1   "offset_pSTAT5 + scale_pSTAT5 * pSTAT5
...
```

- The 1st argument specifies the unique identifiers (here `tSTAT5_au` and `pSTAT5_au`). Please note that this identifier should be the same as the corresponding column header in the data sheet. Also, Observable identifiers are not allowed to be the same as dynamic variable identifiers.
- The 2nd-4th argument specify the unit types (here `C` = concentration), the unit itself (here `"au"`), and a plain text used for plots (here `"conc."`).
- The 5th argument corresponds to a flag (0=no, 1=yes) that indicates if the maximal value in the data sheet of the respective observable should be rescaled to one.

- The 6th argument corresponds to a flag (0=no, 1=yes) that indicates if both the raw data form the spread sheet and the model observables should be compared on a log-10 scale. This is frequently used for concentration data that contains log-normal measurement noise. The transformation is applied after rescaling in case of argument 5 applies.
- The 7th argument specify the mathematical expression of the observable function (here `"scale_tSTAT5 * (STAT5 + pSTAT5)"` and `"offset_pSTAT5 + scale_pSTAT5 * pSTAT5"`. It will be places inside the log-10 of argument in case of argument 6 applies.

1.9 Defining the magnitude of the measurement noise in the ERRORS section

In the likelihood function, the measurement noise is modeled as normal or log-normal distribution, see 2.2 argument 6. The magnitude of the measurement noise, i.e. the standard deviation of the normal distribution (or standard deviation of the normal distribution in the log-space for log-normally distributed noise), can be implemented as parameterized function

```
ERRORS
tSTAT5_au       "sd_STAT5_au"
pSTAT5_au       "sd_STAT5_au"
...
```

- The 1st argument corresponds to an unique identifiers as given in the observables section (here `tSTAT5_au` and `pSTAT5_au`). It is mandatory to specify the measurement noise for each observable in the data definition file.
- The 2nd argument specify the mathematical expression of the noise function (here `"sd_STAT5_au"` and `"sd_STAT5_au"` indicates that both measurements have the same measurement noise). Relative measurement noise can be implemented by using the observable identifier, e.g. `"sd_STAT5_abs + sd_STAT5_rel * tSTAT5_au"`.

1.9.1 Defining the measurement noise in the data sheet

Sometimes, instead of using a parametrized function for the measurement noise, one like to directly specify the amount of noise calculated from replicates in the data sheet. Put these value in the data sheet with column header same as the observable name but with the addition `_std` (here e.g. `tSTAT5_au_std` and `pSTAT5_au_std`). Please note that these values will be interpreted as standard deviation of the data not as variance! An example for this usage can be found in the example application `/Example/Swameye_PNAS2003`.

To activate this mode set the value `ar.config.fiterrors = -1`. `ar.config.fiterrors = 1` indicates the standard mode where the error model is used and estimated together with the dynamics. `ar.config.fiterrors = 0` indicates that the error model is used but not estimated, i.e. its parameters are fixed.

For the visual output, an alternative mode is available that shows the measurement noise as error bars on the data (set `ar.config.ploterrors = 1`) rather than noise model around the trajectories (default `ar.config.ploterrors = 0`).

1.10 CONDITIONS section

In this section conditions in term of the model parameters can be specified. All expression that are not uniquely defined above as either compartments, states or inputs will be treated as free parameters.

```
CONDITIONS
kD                 "koff/kon"
init_Epo_EpoR      "0"
STAT5ActJAK2       "STAT5ActJAK2/init_EpoRJAK2"
...
```

- The 1st argument specifies a target parameter that occurred in the model definition above.
- The 2nd argument specifies a mathematical expression that will replace the target parameter in the model (here `"koff/kon"`, `"0"` and `"STAT5ActJAK2/init_EpoRJAK2"`). This can be any mathematical expression of composed of new and old parameters. Take care not to implement recursive conditions. All replacements are executed once and sequentially, so order matters.

1.11 PARAMETERS section

In this optional section the default settings for parameter can be specified. Please note that the intension of this section is to save final parameter values for documentation purpose. During work in progress parameter values should be manipulate "soft" in the MATLAB workspace variables.

2. Data Definition File

Similar to the model definition file experiments have data definition files that contain experiment specific information. A data definition file is a plain text file with the ending `.def` located in the subfolder `/Data` of the current MATLAB working directory. Each data set, stored in the subfolder `/Data` as `.xls` or `.csv` file, must have its own data definition file with the same name but the suffix `.def`. The data definition files are structured in various section:

```
DESCRIPTION
...

PREDICTOR (or PREDICTOR-DOSERESPONSE)
...

INPUTS
...

OBSERVABLES
...

ERRORS
...
```

```
CONDITIONS
...
```

Sections `DESCRIPTION` and `INVARIANTS` are same as in the model definition file, see 1.1 and 1.7. The remaining sections will be explained in the following. As in the model definition file lines in the data definition file can be commented out using `//`.

2.1 PREDICTOR and PREDICTOR-DOSERESPONSE sections

Is used to define the independent variable / the predictor. For the data definition file there are two modes available, `PREDICTOR` and `PREDICTOR-DOSERESPONSE`. For using the `PREDICTOR` mode see description in Section 1.2, the independent variable in this case is usually time. In the `PREDICTOR-DOSERESPONSE` mode one can use an existing `parameter` as additional independent variable

```
PREDICTOR-DOSERESPONSE parameter
t                T    "min"     "time"    0   100
```

In this mode, there has to be a column named `parameter` in the data sheet, besides the first column that encodes the standard independent variable (here time `t`). Please note that formally this alternative mode does not change the results, however the plot of the model fit to the data will be modified. For instance if `parameter` corresponds to the concentration of a certain input, the result will the a dose response plot.

2.2 Defining the INPUTS

Each experiment can have a different input function, e.g. instance implementing a different treatment scheme. Therefore, in the data definition file the input can be redefined for this specific experiment.

```
INPUTS
Epo              "10 + sin(k1*t)"
SAv              "1 + 0.1*t"
...
```

- The 1st argument corresponds to an unique identifiers as given in the model definition file (here `Epo` and `SAv`).
- The 2nd argument specify the mathematical expression of the input function (here `"10 + sin(k1*t)"` and `"1 + 0.1*t"`), see 1.5 for more detailed explanation.

2.3 OBSERVABLES and ERRORS sections

At this point of the model definition files one can specify observables and their corresponding error model that override the default specifications from the model definition file (see Sections 1.8 and 1.9).

2.4 Defining the CONDITIONS for the specific experiment

Like in the model definition file, see 1.10 for detailed explanation, experiment specific conditions can be specified.

```
CONDITIONS
init_Epo_EpoR    "10"
...
```

Please note that these "data" conditions are applied to the model after the "model" conditions have been applied.

2.5 PARAMETERS section

Like in the model definition file, see 1.11 for detailed explanation, the default settings for parameter can be specified also on the level of the data definition file, i.e. for the additional parameter that might be introduced there.